Discovery Engine for Finding Hidden Connections in Prose Comprehension from References

Amal Babour[1], Javed I. Khan[2], Fatema Nafa[3], Kawther Saeedi[4], Dimah Alahmadi[5]
Faculty of Computing and Information Technology
Department of Information Systems, King Abdulaziz University, Jeddah 21589, Saudi Arabia[1, 4, 5]
Department of Computer Science, Kent State University, Kent, OH 44240 USA[2]
Department of Computer Science, Salem State University, Salem, MA 01970 USA[3]

***Abstract*—Reading is one of the essential practices of modern human learning. Comprehending prose text simply from the available text is particularly challenging as in general the comprehension of prose requires the use of external knowledge or references. Although the processes of reading comprehension have been widely studied in the field of psychology, no algorithm level models for comprehension have yet to be developed. This paper has proposed a comprehension engine consisting of knowledge induction which connects the knowledge space by augmenting associations within it. The connections are achieved through the automatic incremental reading of external references and the capturing of high familiarity knowledge associations between prose concepts. The Ontology Engine is used to find lexical knowledge associations amongst concept pairs, with the objective being to obtain a knowledge space graph with a single giant component to establish a base model for prose comprehension. The comprehension engine is evaluated through experiments with various selected prose texts. Akin to human readers, it could mine reference texts from modern knowledge corpuses such as Wikipedia and WordNet. The results demonstrate the potential efficiency of using the comprehension engine that enhances the quality of reading comprehension in addition to reducing reading time. This comprehension engine is considered the first algorithm level model for comprehension compared with existing works.**

***Keywords*—Knowledge graph; ontology engine; text comprehension; text summarization; Wikipedia; WordNet**

I. Introduction

Text comprehension is a form of knowledge acquisition whereby readers interact with text and relate the ideas represented to their knowledge and experiences [1]. Generally, reading a single text does not qualify readers to achieve the required level of comprehension. This is because the comprehension process depends largely on reader knowledge or additional knowledge acquired from external sources. A well written text embeds a set of cues to build up a coherent representation of the text. Although the text normally presents a set of related concepts, this does not qualify readers to achieve complete comprehension. In this paper, the focus is on prose comprehension. Prose is a type of text which includes complex concepts manifesting particular meanings of a specific domain with associations among concepts. As prose comprehension is difficult to achieve from a simple reading of the text alone, external knowledge is required to understand the text. The external knowledge readers need is known as prior knowledge [2]. This prior knowledge is different from one to one comprehension. Sometimes, readers may not even have a minimum level of prior knowledge about a specific topic, making it necessary for them to obtain it by using external sources. Lexicons and external references are two common examples of acquiring additional knowledge. By drawing upon prior knowledge, readers fill the knowledge gap in the prose by connecting the prose contents with their prior knowledge. The process of prose comprehension is flexible to implement and incorporates external information sources into the prose text. An example of such a process is integrating an encyclopedia with linguistic information/ dictionaries.

Although human readers tend to consult external knowledge mediums such as books, Wikipedia, or journal articles to bridge this knowledge gap, such a process is often time consuming and tedious. It requires reading a large amount of text and then finding the relevant portions of the reference to catalyze understanding. While the mental process behind such knowledge induction is intriguing, adopting computational algorithms can help more effective reading comprehension of prose by automatically capturing relevant text pieces from external references and relevant knowledge association of the prose concepts as a summary. This can leverage greater overall comprehension process of a prose text.

The topic of the human reading rate has been widely studied. Carver's reading model identifies the relationship model between reading and comprehension [3]. He defines five reading processes: memorizing, learning, rauding, skimming, and scanning. The reading rate of an individual varies on the basis of material difficulty and reading objective [3, 4]. For example, skimming may be used when a reader requires an overview of the material, whereas learning may be used when a reader requires comprehension of the material. Carver's reading model provides a reading rate that measures how many words are read per minute (wpm) for each of the five processes. For memorization, the rate is 138 wpm, whereas for learning it is 200 wpm, and for rauding the rate is 300 wpm. For skimming, the rate is 450 wpm, and finally for scanning it is 600 wpm. For example, if readers read a text of 2000 words for the purpose of learning, it takes about 10 minutes for them to read and comprehend it.

The previous work focused on generating summarized texts and/or recruiting participants to analyze their level of comprehension. To the best of knowledge, no existing work proposing an algorithm level model for comprehension. Accordingly, the contribution of this paper can be summarized as follows:

- Develop an algorithm level model comprehension engine to enhance prose comprehension.
- Evaluate the proposed model through set of graph metrics.

The proposed comprehension engine contributes to the reading rate that readers may need to acquire specific knowledge from reference texts. This engine is based on the Knowledge Induction Process which targets increasing knowledge comprehension through two steps: 1) the incremental reading of external reference texts and giving an extractive summary of each by capturing of the highest familiarity knowledge associations amongst prose concepts. One of the distinct features of the proposed algorithm is that it captures the highest familiarity knowledge along with the fewest associations pertaining to prose concepts through a minimum number of external concepts; and 2) using Ontology Engine to find lexical knowledge associations amongst concepts. This can save readers time and increase their efficiency, which are two main advantages of the comprehension engine.

The rest of the paper is structured as follows: Section II provides an overview of the related work. Section III introduces the development of the comprehension engine. Section IV presents an evaluation of the proposed engine. Section V presents the materials used in the experiment and the results, and the final section (Section VI) discusses the conclusion and the future work.

II. Related Work

Considerable research has been done in text summarization and text comprehension in recent years. Some of the research introduced valuable techniques used to produce extractive and abstractive summarization. Extractive summarization summarizes the text by extracting sentences containing salient information from the text itself, while abstractive summarization summarizes the text by paraphrasing the text using words that might not in the text [5]. In recent years, various approaches have been developed for automatic summarization and have been applied widely in different domains.

Van Lierde and Chow [6] combined fuzzy and statistics approaches to obtain extractive summarization. The fuzzy based technique contained manually generated rules where the rules were proceeded based on the length of the sentence. By using these rules, all the sentences were assigned with a weight value ranging from zero to one, where the weight for each sentence was used as a feature in the fuzzy inference system. The authors used such a system to perform the summarization through a number of fuzzy-logic-based analyzers. The work had an advantage of taking into consideration the linguistic variables and human perception.

Hernández-Castañeda et al. [7] proposed a method for extractive automatic text summarization (EATS). The method based on the conversion of the text into numerical vectors by applying different generation methods. The vectors are grouped into clusters based on measuring proximity among the vectors. Latent Dirichlet allocation (LDA) was used to obtain the key sentences in each cluster that make up the summarized text.

Furthermore, Azadani et al. [8] used graph-based summarization techniques to produce extractive summarization. The technique represented a document as a graph in which a node can correspond to various semantic units including words, phrases, concepts or sentences, whereas an edge demonstrated the relation in connectivity between the nodes. The authors used the frequent itemset mining algorithm to extract the summary.

For summarizing multiple documents, Fuad et al. [9] used two techniques, sentence clustering and neural sentence to produce abstractive summarization. In sentence clustering techniques, they used a deep neural network architecture to represent text. In neural sentence, they applied seq2seq encoder-decoder technique. The proposed method took a related ordered set of sentences and produced a single sentence by merging the input sentence from multiple documents; the output was only one sentence as a summary.

For the purpose of text summarization and comprehension, Ding et al. [10] proposed a text summarization generation model to enrich representing the information of the original text and improve the text comprehension. Their method was based on seq2seq through a dual-encoder, the Gain-Benefit gate for decoding, and the probability distributions of the keywords in the text.

In [11], Caglieroet et al. proposed a method named TESTdriven SUMmarization (TestSumm) to recommend text summarization based on a learner's level of comprehension. The method provides multiple-choice tests to assess the learner's comprehension level of different topics. It performs a Multilingual Weighted Itemset-based Summarizer (MWISum) that relies on frequent itemset mining sentence selection and ranking to generate the material summary. It recommends a personalized summary to learners who did not pass the multiple-choice test.

All the previous text summarization research focused on reducing the volume of the text by capturing or rephrasing the most important sentences that include main keywords and present it to the reader. On the other hand, the current text comprehension research based on recruiting participants and applying descriptive statistics to analyze their performance. Thus, this paper has introduced a text summarization method that may increase the number of concepts in the summary in some cases for the purpose of text comprehension.

III. Knowledge Induction Process

"Knowledge graph is an emerging technology for massive knowledge management and intelligent services in the big data era" [12]. It provides an ideal technical solution to realize the integration of knowledge sources by incorporating noisy information and connecting the fragmented pieces of knowledge from multiple sources in a consistent way. In this paper, a knowledge graph called the Illuminated Knowledge

Graph (IKG) is proposed to illuminate the relationships among prose concepts using multiple knowledge sources for the purpose of comprehension. These ideas and goals were the inspiration for the graph name.

The IKG is a graph which captures the state of learning process. It shows prose concepts (CL) and associations amongst the concepts. The associations can be found by reading the prose (LTX) and reading some parts relevant to the concepts as a summary from external reference texts (RTX) as well as an Ontology Engine (OE). Beside the CL concepts, the graph could include external concepts belong to RTX or OE in the associations added from them to connect the prose CL concepts. A directed graph presents IKG = (C, E), where C is a concept set and E is an edge set. A concept c_i includes set of senses ($s_{i,1}$, $s_{i,2}$, .., $s_{i,x}$), where i is the number of concept and x is the number of senses. An edge connects two concepts through a specific sense of each concept. The edge represents a sentential relation between the concepts. It can be a syntactical explicit and/or an Ontology Engine relation, where the latter is one of the six types of word relations: Hyponym, Hypernym, Holonym, Instance, Meronym, or Synonym. Fig. 1(a) illustrates an example of five concepts in IKG. c_1*={Country}*, c_2*={State}*, c_3*={freedom}*, c_4*={Ohio}* and c_5*={Liberty}*, where c_1, c_4 and c_5 are the CL belonging to LTX. c_2 belongs to a RTX. c_3 belongs to an OE.

A Knowledge Path (K) represents the relationship path between two concepts. It is represented as a sequence of edges connecting concepts c_i and c_j in a preserved sense. The concepts c_i and c_j belong to CL. The concepts in the middle may be external to CL. The following are examples of geometric paths:

1) {c_1–$s_{1,1}$: Synonym: $s_{2,1}$–c_2–$s_{2,1}$: Instance: $s_{4,1}$– c_4}.
2) {c_1–$s_{1,1}$: Synonym: $s_{2,1}$–c_2–$s_{2,1}$: syntactic explicit: $s_{4,1}$–c_4}.
3) {c_1–$s_{1,1}$: Synonym: $s_{2,1}$–c_2–$s_{2,2}$: Hyponym: $s_{3,1}$–c_3–$s_{3,1}$: Hyponym: $s_{5,4}$ –c_5}.

c_i refers to the ith concept. $s_{i,j}$ is the jth sense of the ith concept. For example, c_1 is the first concept and $s_{1,1}$ is the first sense of the first concept. Fig. 1(b, c, d, e) represents examples of geometric paths. The knowledge paths can be derived from geometric paths. For example:

1) {c_1–$s_{1,1}$: Synonym: $s_{2,1}$–c_2–$s_{2,1}$: Instance: $s_{4,1}$–c_4}.
2) {c_1–$s_{1,1}$: Synonym: $s_{2,1}$–c_2–$s_{2,1}$: syntactic explicit: $s_{4,1}$–c_4}.
3) {c_2–$s_{2,2}$: Hyponym: $s_{3,1}$–c_3–$s_{3,1}$: Hyponym: $s_{5,4}$–c_5}.

The three paths that mentioned above are considered knowledge paths, as the incoming and the outgoing senses of each concept are preserved. Fig. 1(b, c, e) represents examples of knowledge paths.

A prose can be often rich with concepts using domain specific terms. While reading such prose, non-specialist readers find these terms difficult to comprehend. They may face these difficulties while reading prose in any domain such as science, medical, finance and technology. Therefore, external knowledge helps the comprehension of prose by the integration with the prose concepts [13, 14]. This is not always a straightforward process, as readers' prior knowledge may vary. Inexperienced readers may get overwhelmed by the amount of diverse external knowledge sources and types available since the latter can range from reference texts, dictionaries, and papers to conversations with experts. Identifying the right source of knowledge from a vast range of information can be a time consuming and exhausting task.

Therefore, this work proposes the Knowledge Induction Process to enhance prose comprehension. The process is designed to capture knowledge missing in a prose. The external knowledge source is captured through augmenting knowledge associations of prose concepts using reference texts and an Ontology Engine. The proposed solution assumes that concepts are known prior to the process of finding the knowledge associations amongst the concepts. The process reads appropriate parts from relevant reference texts and then captures a summary of the highest familiarity knowledge associations connecting the prose concepts. The Ontology Engine captures lexical knowledge associations for every concept pair. This process is formally presented in the following way: Given a prose LTX for comprehension, a set of prose concepts CL, CL= {c_i, c_{i+1} ⋯, c_n} ∈ LTX, a set of reference texts RTX, and an Ontology Engine OE. Find the IKG to represent knowledge associations amongst CL. The IKG is built through two fundamental techniques, namely a generation concepts representation and a reference consultation.

A. Generation of Concept Representation

To understand a prose text, a reader may break it up into a set of concepts and then find knowledge associations amongst these concepts. In order to find new or missing knowledge associations among the prose concepts, this process can be automated by applying a computational representation model. It is assumed that this enhances the prose comprehension. A graph is used to represent the concept and the associations, where each concept is represented by a node and each sentential relation between two concepts is represented by an edge.

A Syntactical Explicit Graph generator function KG() is used to convert a prose text LTX to a knowledge graph G_0, and a reference text RTX to a reference knowledge graph GR. For each sentence in LTX/RTX, the function searches for pairs of concepts c_i and c_j if a word or a sequence of words found e_b, b=1,2,…,n between them in the same sentence, where in LTX, c_i and c_j belong to CL and in RTX, c_i and c_j belong to the reference text noun concepts. The distance between c_i and c_j is <= L, where L is the maximum number of words allowed between c_i and c_j. If it does, the function saves [c_i, e_b, c_j] to be an edge in the graph representing a syntactical relation between c_i and c_j.

To allocate the most familiar knowledge associations that connect the prose concepts, it is crucial to evaluate the familiarity of knowledge associations. These can be calculated through the edge weight in the knowledge graph. Calculating the weight/familiarity value is based on the type of the sentential relation between concepts. The sentential relation types reflect the structures between any pair of concepts. The weight $w_{i,j}$ is calculated by (1) where $f_{i,j}$ is the frequency of the relation type between concept c_i and c_j attained from the “Gutenberg Project” [15]. The "Gutenberg Project " is an online resource offering over 53,000 free e-books with expired US copyrights. This online archive is a very popular dataset frequently applied in text mining research [16, 17].

$$w_{i,j} = -1/\left(\frac{1}{log\left(\frac{f_{i,j}}{10^9}\right)}\right) \qquad (1)$$

The familiarity value of a sequence of words between c_i and c_j is calculated based on the lowest weight of words sequence. The log of a word frequency is divided by 10^9, as word frequencies are in the millions. The result is multiplied by -1 to avoid negative values. High frequency refers to high familiarity of a relation type. An inverse relationship between f and w indicates that the higher the frequency, the less its weight or the less its cost.

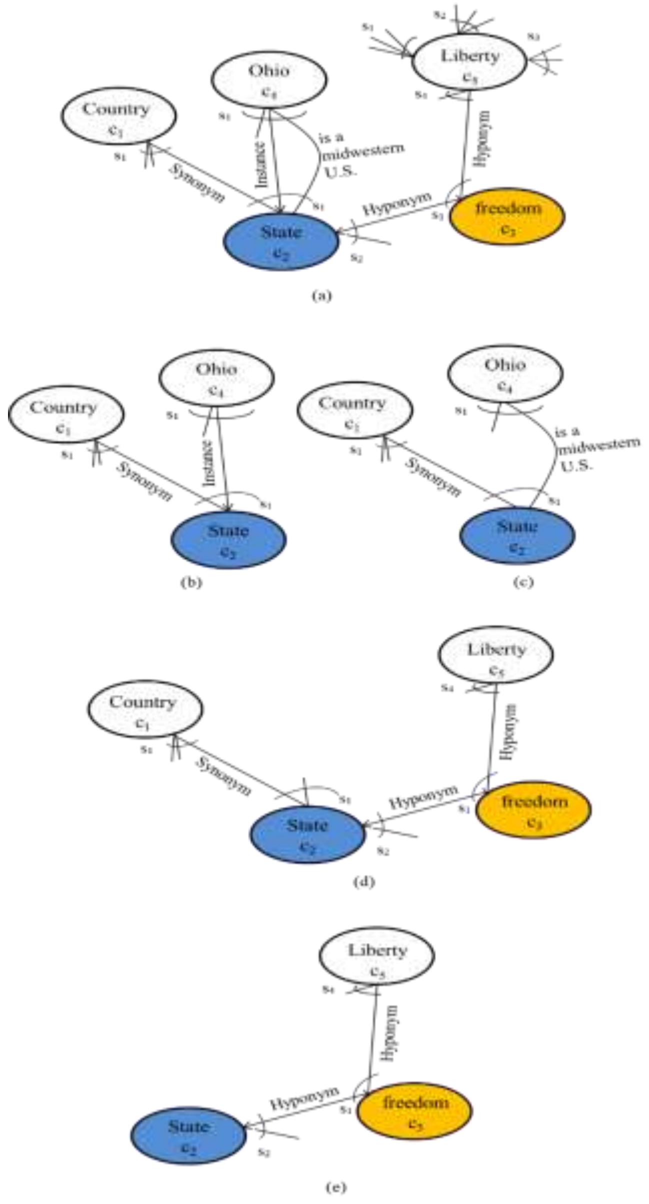


Fig. 1. Example of an Illuminated Knowledge Graph. (b, c, d, e) Exemplifying Geometric Path. (b, c, e) are Examples of Knowledge Paths.

Table I explains five types of the sentential relations between concept pairs. (1) is where a single word e_b appears between c_i and c_j, b=1. (i.e. Carbon-Carbon releases carbonization), here c_i = *Carbon-Carbon*, c_j = *carbonization*, and e_1= *releases*. The weight of the edge is identified by considering the frequency within (1) as the frequency of the word e_1. (2) is where multiple words e_b appear between c_i and c_j, b > 1 (i.e. Ethane is structurally the simplest hydrocarbon), where c_i = *Ethane*, c_j = *hydrocarbon*, b=2, e_1= *structurally*, and e_2= *simplest*. The weight of the edge is computed by considering the weight of e_1 and e_2 separately and then allocating the minimum weight. (3) is a class/subclass, wherein the sentential relation is either a hypernym or hyponym (i.e. Ethane is a hyponym of hydrocarbon), c_i = *Ethane*, c_j = *hydrocarbon* and *e*= *hyponym*. The weight of the edge is determined by considering the frequency within (1) as the frequency of the word class. (4) is a part or subpart, wherein the sentential relation is either a holonym or meronym (Hydrogen is a holonym of water), c_i = *Hydrogen*, c_j, = *water* and *e*= *holonym*. The weight of the edge is determined by considering the frequency within (1) as the frequency of the word part. (5) is a synonym, if the sentential relation is synonym (Ethane is a synonym for C_2H_6). c_i = *Ethane*, c_j = C_2H_6 and *e*= *synonym*. The frequency of the synonym relation is supposed to be 1. Therefore, the weight of the edge is determined by considering 1 in the frequency given by (1).

To generate an IKG, the Knowledge Induction Process performs six steps as presented in Fig. 2:

1) Converting a prose LTX to a prose knowledge graph G_0 representing the syntactical relation between each pair in the prose concepts CL by performing a Graph generator function KG().

2) Converting a reference text RTX_i to a reference knowledge graph GR_i representing the syntactical relation between each pair of the RTX_i concepts by performing the same Graph generator function KG() used in step 1.

3) Extracting the highest familiarity knowledge path(s) of RTX_i connecting CL from GR_i by performing the Terminal to Terminal Traffic Steiner Tree function TTTST(). The extracted path(s) is called a Terminal to Terminal Traffic Steiner tree(s) TTTST and it is represented in a graph called GU_i.

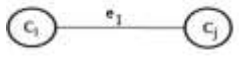
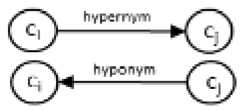
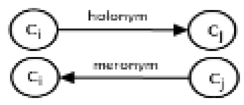
4) Joining G_0 and GU_i in a graph called Gtemp representing the current state of the assimilated knowledge among CL by performing an assimilation function Gassmilation().

5) Finding OE-Knowledge-Path(s) connecting each pair of Gtemp concepts by performing the OE-Knowledge-Paths function KPOE(). The found paths are represented in a graph called GW_i.

6) Joining Gtemp and GW_i in an Illuminated Knowledge Graph IKG_i representing the current state of the assimilated knowledge amongst CL by performing the assimilation function Gassmilation().

Each time the process reads a new RTX_i, it performs steps 2 to 6 where, in step 4, G_0 is replaced with IKG_i. Fig. 3 shows an example explaining the impact of using reference texts and an Ontology Engine for adding knowledge associations amongst a set of prose concepts about 'Ethane' chemical compound, where CL= {*ethane*, *hydrocarbon*, *hydrogen*, *carbon*, *carbon-carbon and carbonization*}.

TABLE I. TYPES OF SENTENTIAL RELATIONS

Sentential relation type	**Sentential relation structure**	**$w_{i,j}$ value**
Syntactical Relation		
	(1): Single word: $c_i - s_{i,*} : e_b : s_{j,*} - c_j$; b=1	$w_{i,j} = w(e_1)$
c_i $e_1 \ldots e_2$ c_j	(2): Multiple words: $c_i - s_{i,*} : e_1\ e_2 \ldots e_n : s_{j,*} - c_j$; b=1,2,…,n	$w_{i,j} = \min(w(e_b))$
OE Relation		
	(3): Class/Subclass: $c_i - s_{i,*}$: Hypernym : $s_{j,*} - c_j$ or $c_i - s_{i,*}$: Hyponym : $s_{j,*} - c_j$	$w_{i,j} = w(e_{class})$
	(4): Part/Subpart: $c_i - s_{i,*}$: Holonym : $s_{j,*} - c_j$ or $c_i - s_{i,*}$: Meronym : $s_{j,*} - c_j$	$w_{i,j} = w(e_{part})$
c_i synonym c_j	(5): Synonym: $c_i - s_{i,*}$: Synonym : $s_{j,*} - c_j$	$w_{i,j} = w(e_{synonym}) = 1$

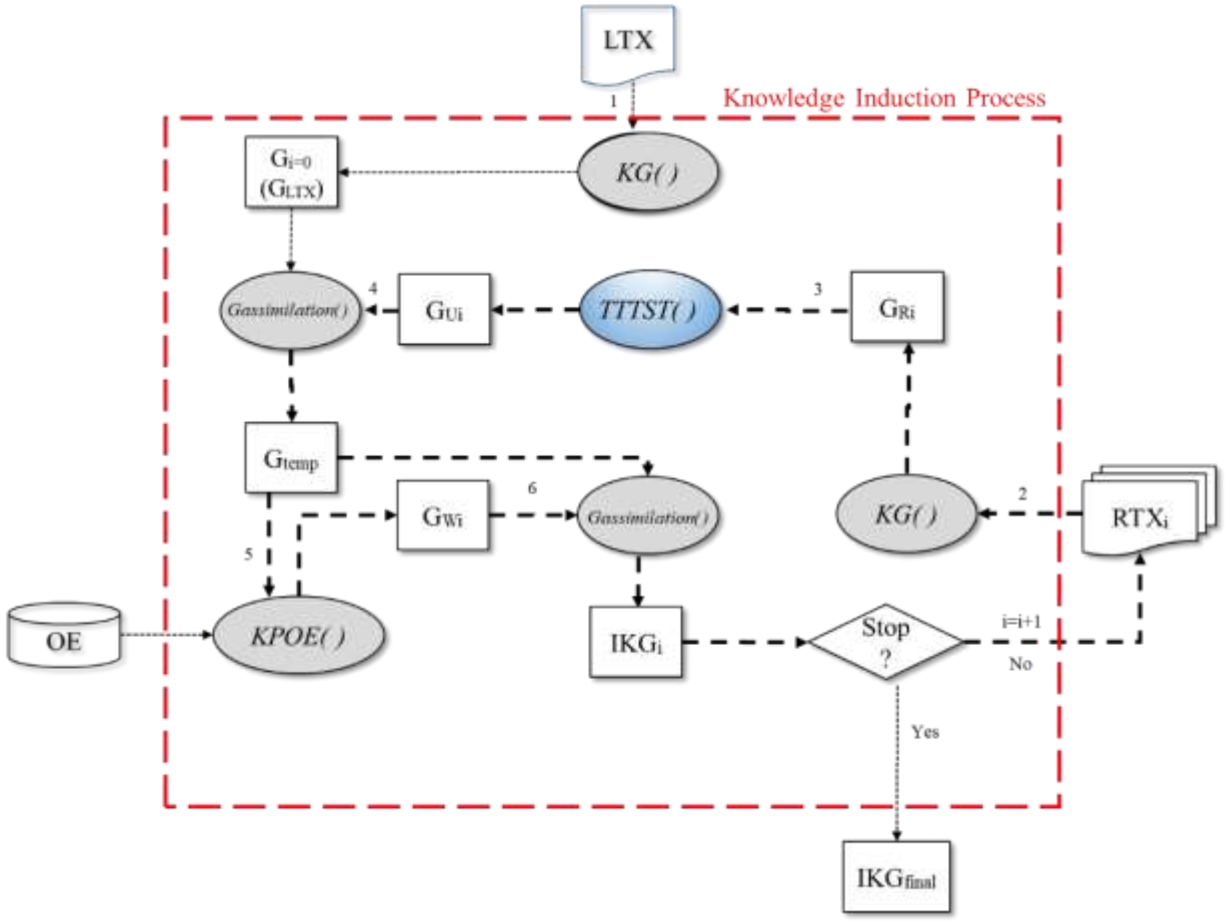


Fig. 2. Comprehension Engine.

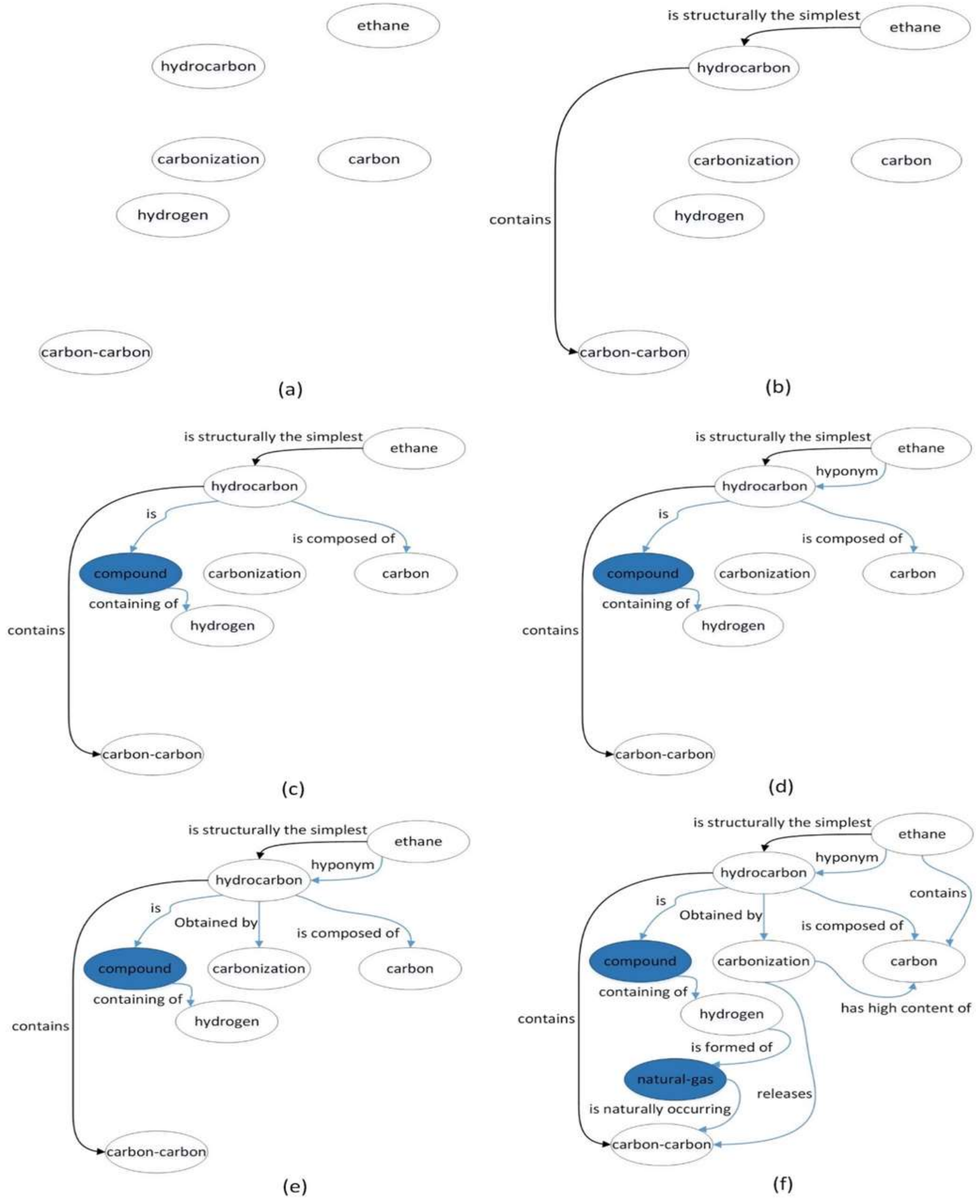


Fig. 3. Connecting CL Process using Reference Texts and Ontology Engine. (a) a set of Prose Concepts at the Initial Prose. (b) Knowledge Path K from the Initial Prose LTX. (c, e, f) Additional Knowledge Path K Extracted from RTX1, RTX2, and RTX3. (d) Additional Knowledge Path K Extracted from Ontology Engine OE.

B. Reference Consultation

1) Highest familarity knowledge extraction using a reference text: Knowledge within a prose is limited, as readers cannot capture the knowledge relevant to all concepts required to comprehend the prose within the prose text only. Therefore, external reference texts are required to capture knowledge association amongst prose concepts. The reference texts are used to fill the knowledge gap of prose.

There are a number of Steiner tree versions that can be useful for identifying knowledge associations amongst the prose concepts [18, 19]. The main role of a Steiner tree in the proposed engine is to capture the highest familiarity knowledge paths amongst the prose concepts CL from each reference text GR_i with a minimum cost and a minimum number of external concepts, where the sentences of the captured knowledge paths are considered the summary of the reference text.

Given a connected, undirected reference knowledge graph $GR_i= (C, E)$, where C is a set of reference text concepts and E is a set of edges representing the relations amongst the graph concepts, the weight for each edge $w_{i,j}$ reflects its cost, where the cost here expresses the familiarity value of the sentential relation between the concepts of the edge ends c_i and c_j as shown in Fig. 4(a). Fig. 4(b) shows the set of prose concepts CL that need to find the highest familiarity knowledge paths amongst them in GR_i. Minimum Steiner Tree (MST) is an approach based on finding knowledge with the minimum cost amongst prose concepts [18]. The cost of MST is calculated by $\sum(w_{i,j})$, where $i, j \in C$, $i \neq j$. Sometimes, connecting the CL in a MST requires the addition of external concepts to CL. This case can be found in Fig. 4(c), C= {A, B, C, D, E}, CL= {A, C, E, D}. The returned Steiner tree is {E, D, B, A, C} and it costs 10.

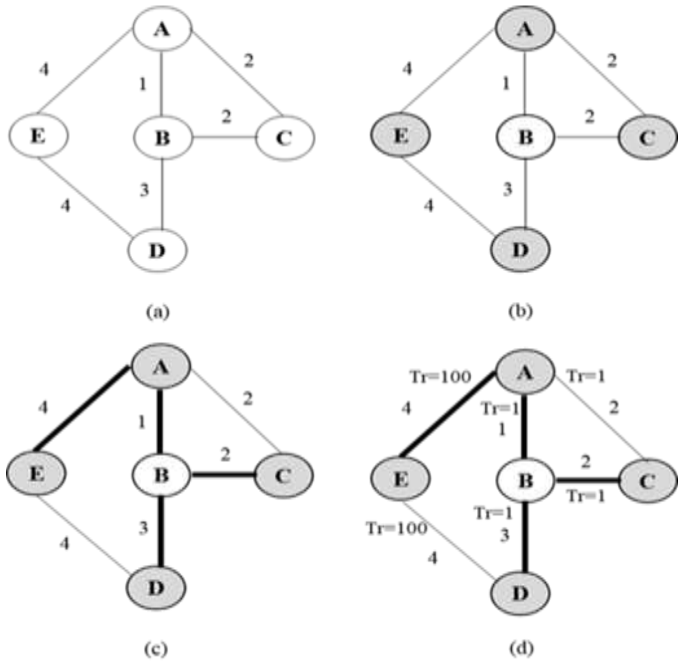


Fig. 4. Example of a Reference Knowledge Graph and different Types of Steiner Trees.

Suppose there is traffic (Tr) between c_i and c_j, where the traffic indicates the comprehension amount of the relation between the two concepts. In this case, the aim of the Steiner tree is to reduce the traffic amongst CL, whereby low traffic means high comprehension. Here, the Steiner tree is called Terminal to Terminal Traffic Steiner Tree (TTTST). Its cost is $\sum(w_{i,j} \times Tr_{i,j})$, where $i, j \in C$ and $i \neq j$. In Fig. 4(d), supposes that the traffic between each two concepts is $Tr_{i,j}=1$ and the traffic between concept E and concept A is $Tr_{E,A}=100$ and between concept E and concept D is $Tr_{E,D}=100$. In this case, the traffic weighted cost for the returned Steiner tree is $((4 \times 100) + (1 \times 1) + (3 \times 1) + (2 \times 1)) = 406$.

Fig. 5 represents the TTTST algorithm in the following way. The algorithm input is GR_i and CL. GR_i is a reference knowledge graph representing the syntactical relation between each pair of RTX_i concepts. The algorithm output is GU_i, which is a tree extracted from GR_i representing the highest familiarity knowledge path(s) connecting CL.

```
Def TTTST ( ):

Input: GR_i, CL
Output: TTTST: Terminal to Terminal Traffic Steiner Tree

1. //initialization
1. for each co in GR_i:
3.     if CL!= null:
4.      for each concept c in co
5.      prev[c]= -1
6.      cost[c]=INFINITY
7.      Visited[c]=False
8.      Q=null
9.      s= pick any concept from CL
10.     enqueue(Q,s)
11.     While Q!= null:
12.     c= dequeue(Q)
13.     Visited[c]=True
14.     if c in CL:
15.     cost[c]= 0
16.             add c to M
17.             remove c from CL
18.     for each neighbour c_i of c:
19.     if c_i not in Q and Visited[c_i]==False:
20.             enqueue(Q, c_i)
21.             temp= cost[c] + w_ci,c
22.             if temp == cost[c_i] and c in CL:
23.     prev[c_i]=c //a knowledge path with minimum external concepts
                        to c_i is found
24.             cost[c_i]=temp
25.             else if temp < cost[c_i]
26. prev[c_i]=c // a less costly knowledge path to c_i is found
27.             cost[c_i]=temp
28. TTTST = getPaths(M[ ], prev[ ])
29. return TTTST
```

Fig. 5. Terminal to Terminal Traffic Steiner Tree Algorithm.

For each component *co* in GR_i, the algorithm uses a queue data structure *Q* to store each visited concept with its neighbors. It initializes *Q* by picking any concept from CL as the source *s*. Then, it initializes the cost between *s* and each concept *c* in the *co* to INFINITY and the previous concept *prev* of each *c* to -1. In the loop iteration, it dequeues the first concept *c* in *Q*, marks it as visited, and checks if *c* belongs to CL. If so, it updates its cost to 0, adds it to *M* where *M* stores the found CL concepts, and removes it from CL. Then, it enqueues all the neighbors' c_i*'s* of concept c if they are marked as non-visited, assigns *prev* and calculates the cost for each of them. If the current c_i cost is equal to the previous one and c belongs to CL that means a knowledge path with minimum external concepts is found. Then, c_i*'s prev* is updated to the new concept. However, if the current c_i cost is less than its previous one that means a less costly knowledge path to c_i is found wherein less cost means high familiarity. Then, c_i*'s prev* and cost are updated to the new lesser values and the process is repeated until the queue is emptied. If all items in *co* are checked, getPaths() constructs the TTTST from *M* and *prev*. The returned TTTST is represented in GU_i [20]. Suppose the graph shown in Fig. 6 is a reference knowledge graph GR_i. If CL = $\{c_1, c_7, c_8\}$ and the traffic for each edge $Tr=1$, then the TTTST returned by algorithm will be $\{c_1,c_3,c_6,c_7,c_8\}$.

2) Knowledge extraction using ontology engine: A useful source of knowledge is the Ontology Engine. An Ontology Engine to provide lexical knowledge associations between any two concepts by using OE-Knowledge-Paths function KPOE() is utilized. This helps in adding a new type of knowledge which can contribute to increasing knowledge and improving comprehension. The OE-Knowledge-Path is a sequence of edges connecting any two concepts in a preserved sense. Each edge represents an ontological relation between its ends, where the ontological relation for each edge is one of the following relations: (1) synonym; (2) hypernym; (3) hyponym; (4) holonym; and (5) meronym.

Fig. 7 represents the KPOE algorithm. It uses an Ontology Engine to search for OE-Knowledge-Path(s) connecting concept *s* and concept *t*, where *s* and *t* are the first and the last concepts in the path and the length of the path is less than or equal to α. The input of the algorithm is *s*, *t*, *R*, and *relationalGraph*. *R* is a dictionary that holds all the ontological engine relations in the Ontology Engine, and *relationalGraph* is a dictionary of all concepts that have any of the ontological relations in *R* with the last node of the current path. The algorithm does not return the shortest path between *s* and *t* because it could be a path of multiple sense concepts. Rather, it searches if the neighbors of *s* have the same sense of *s* and have any of the ontological relation in *R* with *s*. If so, it searches the neighbors of the neighbors until it reaches *t*.

NodeQ and *PathQ* are two queue data structures used in the algorithm. *NodeQ* keeps the current path that has a concept need to explore its neighbor. *PathQ* holds the currently created paths. The algorithm starts with *s* as the current path. The while loop iterates through *PathQ* paths searching for an OE-Knowledge-Path that connects *s* and *t*. In each iteration, it dequeues the first path in *PathQ* and signs it in *NodeQ*. Then, it checks whether the last concept in *NodeQ* matches *t*. If so, it saves the path in *Kpaths* as an OE-Knowledge-Path between *s* and *t*; otherwise, it checks if the *NodeQ* length is less than α. If the last concept in *NodeQ* matches *t*, for the sense of the *NodeQ's* last concept, the function gets all of the concepts that have one of the ontological relations in *R* with the last concept in *NodeQ*, and adds them to relationalGraph. A number of paths are created between each relationalGraph concept and the current path. *PathQ* saves the newly created paths. If all the paths in *PathQ* are checked and *Kpaths* do not exist, the function returns 'Not found' [20].

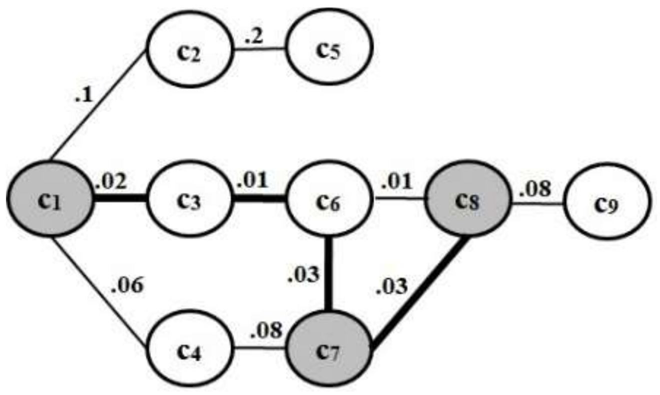


Fig. 6. Terminal to Terminal Traffic Steiner Tree Example.

Def KPOE ( ):

```
Input: s, t, R, α
Output: OE-Knowledge-Paths between s and t
PathQ = [ ]
1.  Kpaths = [ ]
2.  // push the first path into PathQ
3.  PathQ.append([s])
4.  for sen in s.sense( ):
5.   while PathQ:
6.   // get the first path from the PathQ
7.   NodeQ = PathQ.pop(0)
8.   // get the last node from NodeQ
9.   node= NodeQ[-1]
10.  // path found
11.  if node == t:
12.  Kapths.append(NodeQ)
13.  return Kpaths
14.  else:
15.  If len(NodeQ) <= α:
16.  sl= list( )
17.  for key, value in R.iteritems( ):
18.  re= value
 // get all concepts have relations from R with node and have the same sense
of node
20.  x= re (node, sen, key)
21.  sl=sl+x
22.  relationalGraph[node] = sl
// enumerate all adjacent nodes, construct a new path and push into the queue
24.  for adjacent in relationalGraph.get(node,[ ]):
25.  new_path=list(NodeQ)
26.  new_path.append (adjacent)
27.  if len(new_path) < α:
28.  PathQ.append(adjacent)
29.  else:
30.  break
31. if !(Kpaths):
32.  return 'Not found'
```

Fig. 7. Discovering OE- Knowledge-Paths Algorithm.

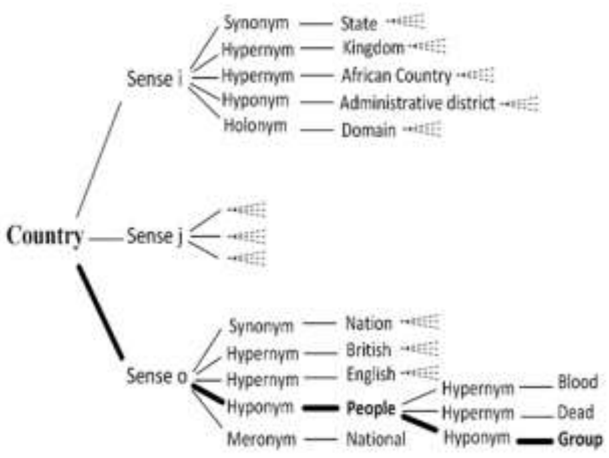


Fig. 8. Example of an OE-Knowledge-Path.

For example, consider *s=“country”*, *t=“group”*, and *α= 4*. The returned OE-Knowledge-Path by the algorithm is *country (hyponym) people (hyponym) group*. The process of discovering the OE-Knowledge-Path between *“country”* and *“group”* is shown in Fig. 8.

IV. Comprehension Model Evaluation

An evaluation model is needed to assess the comprehension engine. The statistical characteristics of the knowledge obtained by the comprehension engine need to be measured. However, the process of measuring knowledge is still a difficult area that needs to be explored. “The fluid and intangible nature of knowledge makes its measurement an enormously complex and daunting task” [21].

The paper has proposed the use of graph theoretical metrics that provide a natural way to describe the structure and interplay of graph edges. A set of graph metrics presented in this section are divided into two types: quantitative assessment and organizational assessment. The mentioned types are important for the following reasons. (1) Quantitative assessment calculates rare knowledge growth, knowledge overload, and entropy to elaborate different aspects about the amount of knowledge that can be gained from the graph. (2) Organizational assessment calculates the size of the giant component, cluster coefficient and graph density to study the strength of the connections between concepts and their neighbors and the connectedness of the graph where high connectivity means high comprehension. The following section explains the real comprehension model based on all the mentioned graph metrics.

A. Content of Information

Comprehension enhancement is influenced by the size of obtained knowledge in the IKG. The obtained knowledge graph size can be computed by counting the entire number of concepts C and the entire number of relations E in IKG. Based on the Knowledge Induction Process, the graph starts with the prose knowledge graph G_0. It is transformed into IKG_1 after reading RTX_1, and then into IKG_2 after reading RTX_2 until it reaches IKG_{final} after reading all the RTX_i. At each state, the size of IKG may be increased and knowledge may be augmented. The knowledge growth rate λ is computed by (2) where $|IKG_i|$ is the IKG size after reading RTX_i and $|G_0|$ is the prose knowledge graph size.

$$\lambda = \frac{|IKG_i|}{|G_0|} \tag{2}$$

The knowledge overload rate *γ* may also be increased, and it can be determined by (3):

$$\gamma = \frac{|IKG_i - G_0|}{|G_0|} \tag{3}$$

The amount of information that can be obtained from the knowledge graph can also be calculated by measuring the Entropy *δ* of the knowledge graph, where high entropy indicates that the knowledge graph has a high amount of information and vice versa. Based on [22], *δ* can be computed by (4):

$$\delta = -\sum_{i=0}^{n} p_i \log(p_i) \tag{4}$$

Where p_i is the degree distribution probability of concept c_i in the knowledge graph that can be determined by (5):

$$p_i = \frac{d_i}{2|E|} \tag{5}$$

Where, d_i is the degree of concept c_i and |E| is the entire number of the relations in the knowledge graph. *δ* is divided by *log(n)* to obtain a measure between 0 and 1, where *n* is the entire number of the knowledge graph concepts.

B. Knowledge Organization

It is obvious that the existence of relationships amongst concepts in the prose influences comprehension. The greater the increase in the relations among the concepts, the higher the understanding of the relations between them. This falls under a graph organization notion that can be calculated by the size of the giant component [23] that can be calculated using (6):

$$GC = \frac{|C`|}{|C\,|} \tag{6}$$

Where |C`| is the number of connected concepts forming a giant component and |C | is the total number of concepts in the knowledge graph. The relation amongst the concepts and their neighbors in the knowledge graph can be calculated by the cluster coefficient *β* [24]. Based on [25], *β* is obtained by using (7):

$$\beta = \sum_{i=0}^{n} \frac{2NIC_i}{d_i(d_i-1)} \tag{7}$$

Where NIC_i is the neighbors’ interconnection coefficient of concept c_i, which denotes to the number of the sentential relations between the first neighbors of concept c_i, and d_i is the sentential relations of concept c_i which counts the first neighbors of concept c_i. The graph is considered highly clustered when the value is towards 1. The knowledge graph density *ρ* is another way to study the Knowledge graph organization to explore the completion and the integrations amongst concepts [26]. A complete knowledge graph contains all possible sentential relations E and density equals 1. The graph density *ρ* is calculated by (8) [27].

$$\rho = \frac{|E|}{|C|(|C|-1)} \tag{8}$$

Calculating the knowledge growth rate λ, the knowledge overload γ rate, and the cluster coefficient β require a reading of the external consultation to perform their calculations. Thus, their values are zero before reading the external references. The following example explains the influence of reading three reference texts with an Ontology Engine for adding knowledge associations amongst the prose concepts. The comprehension model evaluation for the proposed metrics before using external consultations in Fig. 3(b), and after using them in Fig. 3(f), respectively, is shown in Table II.

TABLE II. EXAMPLE OF THE COMPREHENSION ENGINE EVALUATION BEFORE AND AFTER USING REFERENCE EXTERNAL CONSULTATION

	Before using external consultations	**after using external consultations**
Growth λ	0	2.5
Overlap γ	0	1.5
Entropy δ	1.5	0.96
Giant Component Size GC	0.5	1
Cluster Coefficient β	0	0.92
Density ρ	0.07	0.21

V. EXPERIMENT

A. Content Material

An experiment was conducted on three prose texts LTX, and eight concepts were selected arbitrarily as the list of the prose concepts CL from each LTX. Wordnet [28] is a reliable Ontology Engine that has been used by many researchers in this area. It is a vast lexical database introduced by George Miller at the Cognitive Science Laboratory at Princeton University that is used as a dictionary of word senses and semantic relations between words [29-31]. This experiment was based on Wordnet Version 1.7 as a general reference text.

The used benchmark reference text articles for this paper were derived from https://en.wikipedia.org. Wikipedia articles were employed as RTX. For each LTX, a number of articles are taken from Wikipedia to represent each concept in CL. For the article allocation, the method presented in [32] to automatically select Wikipedia articles was chosen. Table III displays the selected LTX_i, as well as the CL for each prose.

For each LTX_i, the sentential relation among the concepts in its CL is represented in the prose knowledge graph G_0. Through the Knowledge Induction Process, the system builds a set of Illuminated Knowledge Graphs IKG_i by scanning all of the RTX, where IKG_i is the sentential relation amongst the concepts in CL after reading a new RTX_i. The complete list of the used reference text articles can be found in Appendix Table V. All the references texts were accessed on July 24, 2019.

B. Results

This section presents the analysis of information gained from the Illuminated Knowledge Graph IKG_i of prose LTX_i in the Knowledge Induction Process.

TABLE III. LIST OF THE PROSES USED IN THE EXPERIMENT

	Prose Title	**List of prose concepts CL**
1st prose LTX_1	'Ethane chemical compound'[33]	['Ethane', 'hydrocarbon', 'hydrogen', 'carbon', 'chemical, 'petroleum', 'carbonization', 'coal']
2nd prose LTX_2	'New Test for Zika OKed' [34]	['Zika', 'infection', 'dengue', 'chikungunya', 'virus', 'aedes', 'mosquito', 'antibody']
3rd prose LTX_3	'Anesthesia gases are warming the planet' [35]	['Anesthetic', 'carbon', 'climate', 'oxide', 'desflurane', 'isoflurane', 'sevoflurane', 'halothane']

The average time needed to read the prose, the eight references attached to the prose and identification of the highest familiarity of sentential relations connecting the prose concepts in each reference are as the following: 4 minutes and 16 seconds for the 1st prose (LTX_1), 5 minutes and 18 seconds for the 2nd prose (LTX_2), and 2 minutes and 7 seconds for the 3rd prose (LTX_3). As noticeable, the machine spends a few minutes to read each prose with its related references.

Fig. 9 displays the information growth λ per knowledge graph in each prose LTX_i. The x-axis represents the prose knowledge graph G_0 and the Illuminated Knowledge Graph IKG_i after reading each reference text RTX_i, whereas the y-axis represents the growth rate λ. In LTX_1, there is a gradual increase in the information after reading reference texts RTX_1 to RTX_6, even though no new information was inserted after reading RTX_7. Interestingly, it started to increase again after reading RTX_8. In LTX_2, the information kept growing gradually from reading RTX_1 to RTX_8. In LTX_3, information increases varied. The information grew after reading RTX_1, while no new information was added after reading RTX_2 to RTX_4. It increased again after reading RTX_5 and RTX_6, although no new information was inserted after reading RTX_7 and RTX_8. This leads to the conclusion that reading references in some cases affect the increase of the knowledge positively, while in other cases these yield no effect.

The information overload rate γ in the knowledge graph is shown in Fig. 10. Here, the x-axis represents the prose knowledge graph G_0 and the Illuminated Knowledge Graph IKG_i after reading each reference text RTX_i, and the y-axis represents the overload rate γ. Similarly, it is clear that information overload varies from being slight to high in the three proses LTX after reading new reference texts RTX_i.

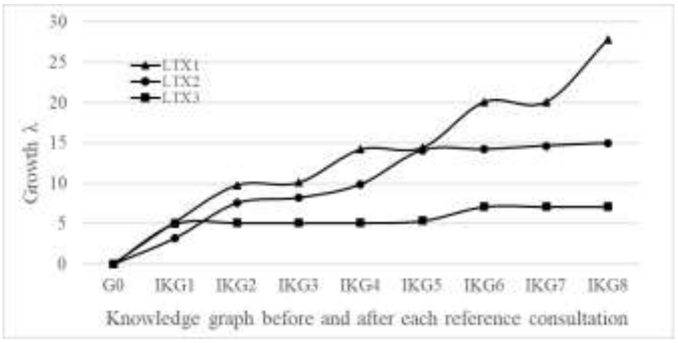


Fig. 9. Information Growth Rate "λ" Per the Knowledge Graphs.

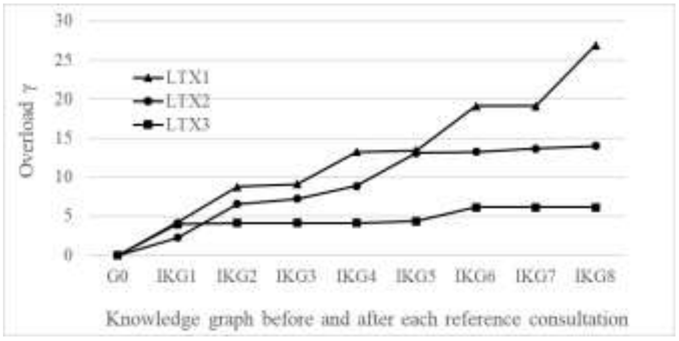


Fig. 10. Information Overload Rate γ Per the Knowledge Graphs.

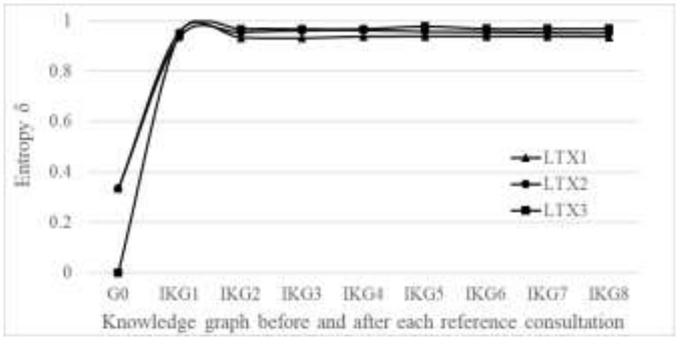


Fig. 11. Entropy and Per the Knowledge Graphs.

Fig. 11 represents the entropy δ per each knowledge graph. The x-axis locates for both the prose knowledge graph G_0 and the Illuminated Knowledge Graph IKG_i after reading each reference text RTX_i, and the y-axis is the entropy δ. It was observed that δ in the three proses LTX started with low values, and then it bounced to high values after reading the 1st reference text RTX_1. This implies that RTX_1 in the three proses LTX contributed to adding a high amount of information. Then, in LTX_1 the entropy value increased a little after having read RTX_2 to RTX_6 and RTX_8. This means that texts RTX_2 to RTX_6 and RTX_8 inserted a low amount of information, while no new information was inserted when reading RTX_7. In LTX_2, there was a little increase in entropy after reading texts RTX_2 to RTX_8; thus, a limited amount of information was inserted. In LTX_3, there was also a slight increase in the entropy after reading RTX_2 and RTX_5; a low amount of information was inserted here, and no new information was added after having read texts RTX_3 to RTX_4, and RTX_6 to RTX_8. This means that some of the reference texts are highly effective in adding information, while other texts have little effect.

Fig. 12 represents the amount of information that was gained due to contributions of the prose LTX, the reference text RTX, and the Ontology Engine OE concepts in the three proses separately. As shown in Fig. 12(a), for LTX_1, the entropy value of OE concepts bounced to a greater value than that of the LTX and RTX concepts after reading texts RTX_1 to RTX_8. This finding suggests that the amount of information inserted by the contribution of OE concepts was higher than the amount inserted by the contribution of the LTX and RTX concepts. In Fig. 12(b), for LTX_2, the highest entropy value of the LTX, OE, and RTX concepts varied after reading RTX_1 to RTX_8; this indicated that the highest amount of information obtained by the contribution of the LTX, OE, or RTX concepts was not concentrated on the contribution of any one of them. In Fig. 12(c), for LTX_3, the entropy value of the OE concepts was the highest amongst the LTX and RTX concepts after having read RTX_1 to RTX_8. This means that the amount of information inserted by the contribution of the OE concepts is higher than the amount inserted by the contribution of the LTX and RTX concepts. This explains the importance of the LTX, RTX, and OE concepts when inserting information, showing that none of these concepts show greater gains than any other of these concepts.

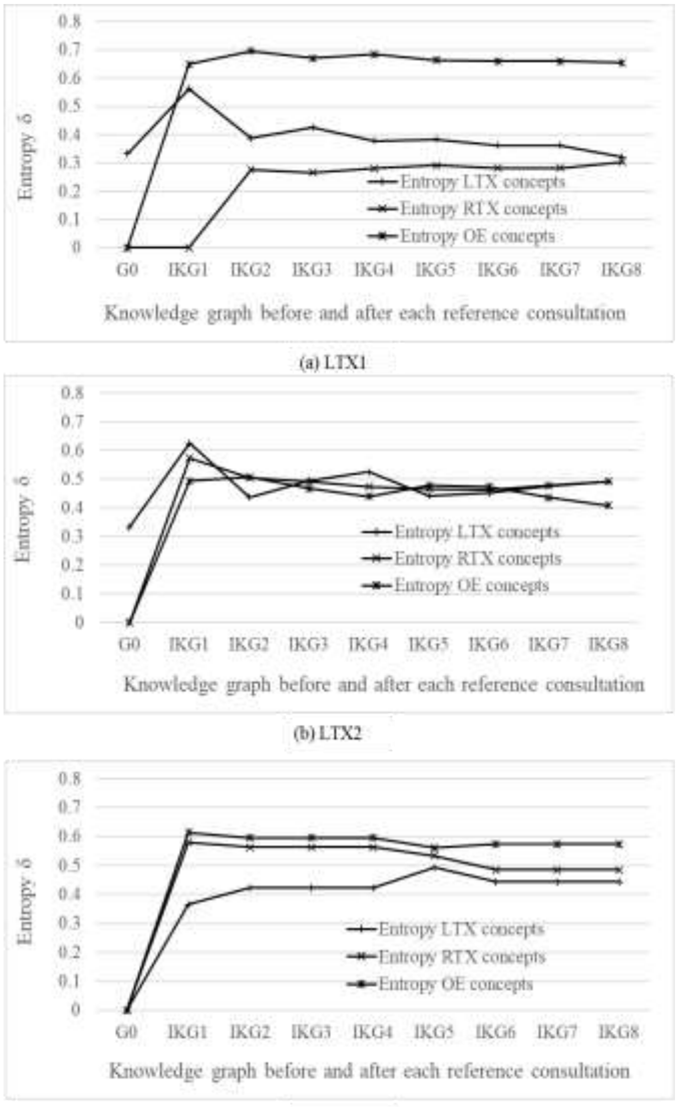


Fig. 12. Break Down of the Entropy and Per the Knowledge Graphs for the Three LTX.

Fig. 13 shows the size of the giant component of the knowledge graph both before and after reading the reference texts RTX_i. The x-axis refers to the prose knowledge graph G_0 and the Illuminated Knowledge Graph IKG_i after reading each reference text RTX_i, and the y-axis is the size of the giant component GC of the knowledge graph. When the process reads a new RTX_i, the size of the giant component increases. It is observable that for LTX_1, the giant component size moved from 0.25 to 0.85 after reading RTX_1, and growing to as high as 1 after reading RTX_8. For LTX_2, the giant component size was 0.25 in the prose knowledge graph G_0, and after reading RTX_1, it reached to 1 and the concepts were fully connected. Meanwhile in LTX_3, the giant component size bounced from 0 to 0.85 after reading RTX_1 and arrived at 1 after reading RTX_3. It means that there was an enhancement of prose comprehension by reading reference texts for connecting the prose concepts in a giant component and illuminating sentential relations amongst them.

The breakdown of the total number of concepts C and the number of the sentential relations E in the prose knowledge graph G_0 and the Illuminated Knowledge Graph IKG_{final} for each prose LTX_i are shown in Table IV, wherein the concepts are from the prose LTX, the reference text RTX, and/or the Ontology Engine OE. As can be seen, there is a great variance in the number of concepts and the number of sentential relations between G_0 and IKG_{final}. Increasing the size may affect the information size positively. Therefore, increasing the IKG_{final} size is a significant sign of the overflowing information in the IKG_{final}, which can further support reinforcement of the prose comprehension.

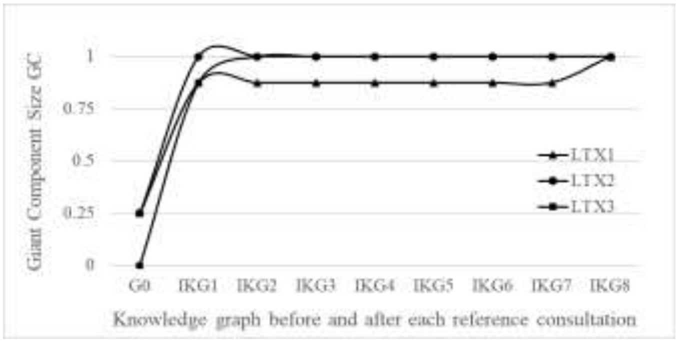


Fig. 13. Giant Component Size GC per the Knowledge Graphs.

TABLE IV. BREAK DOWN OF THE TOTAL NUMBER OF SENTENTIAL RELATION AND CONCEPTS IN THE THREE PROSES

	1st prose LTX_1		**2nd prose LTX_2**		**3rd prose LTX_3**	
	G_0	***IKG_{final}***	***G_0***	***IKG_{final}***	***G_0***	***IKG_{final}***
Sentential relation	1	168	1	90	0	33
Number of prose LTX concepts	8	8	8	8	8	8
Number of reference concepts	0	12	0	16	0	5
Number of Ontology Engine OE concepts	0	63	0	21	0	11

Moreover, Fig. 14 illustrates the clustering coefficient β obtained in each knowledge graph. The x-axis refers to the prose knowledge graph G_0 and the Illuminated Knowledge Graph IKG_i after reading each reference text RTX_i, whereas the y-axis represents the clustering coefficient β. It is clear that some of the graphs are highly clustered; this is due to the fact that many of the concepts within these graphs are highly related to one another. For LTX_1, the graph tended to be highly clustered, especially after reading RTX_1, RTX_4, RTX_5 and RTX_8. In LTX_2, the cluster coefficient bounced to a higher value after reading RTX_2, lowered slightly after reading RTX_3, then slightly increased after reading RTX_4 to RTX_8. In LTX_3, the cluster coefficient, which was higher when reading RTX_1 became stable after reading RTX_2 to RTX_5. It increased again after reading RTX_6 and then returned to being steady after reading RTX_7 and RTX_8. This means that there is enhancement of comprehension when some of the reference texts increase clustering of the concepts. In cases where the texts exercise a limited effect on clustering of the concepts, enhancement of comprehension is not observed.

Fig. 15 gives us a view of the integration amongst the concepts included within each knowledge graph. The x-axis is the prose knowledge graph G_0 and the Illuminated Knowledge Graph IKG_i after reading each reference text RTX_i, and the y-axis is the graph density ρ. As the graph reflects, in the three proses LTX, ρ showed a variance amongst the reference texts when inserting sentential relations amongst the concepts, bringing them close together.

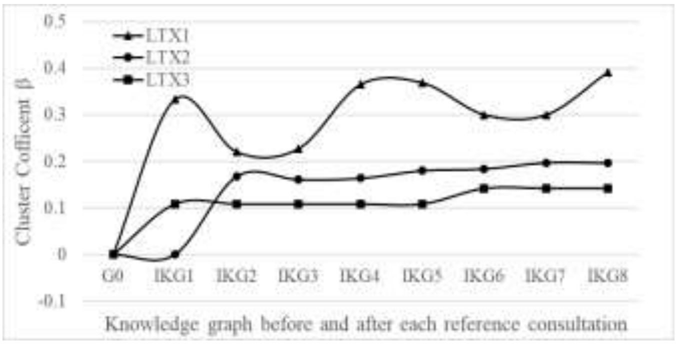


Fig. 14. Cluster Coefficient β Per the Knowledge.

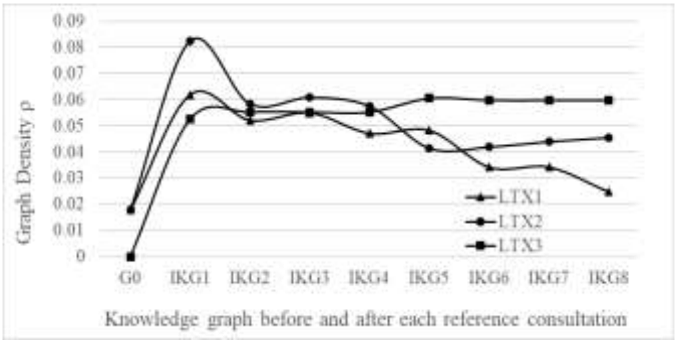


Fig. 15. Density ρ Per the Knowledge Graphs.

VI. Conclusion

The objective of this paper was to enhance prose comprehension by reducing the time expended on understanding the text and increasing the quality of comprehension. Accordingly, the comprehension engine was developed to read several reference texts, to allocate a summary of the highest familiarity knowledge amongst a set of target concepts in each reference and to provide lexical associations among the concepts by using an Ontology Engine. The comprehension engine was evaluated through set of graph metrics. The output performance of the comprehension engine demonstrates the efficiency of the compression engine in reducing reading time and increasing the quality of comprehension. Furthermore, the proposed engine can have academic implications on students or academic learners by providing them with the most familiar and lexical knowledge that can help in prose comprehension in a short time. Although the proposed engine may help readers to increase their knowledge and improve prose comprehension, at times this can causes mess and give rise to 'too much knowledge' in prose comprehension, thus challenging preservation and retention by human memory. Future work will be carried out to develop a second phase of the comprehension engine that efficiently select the less difficult to understand knowledge from the augmented knowledge and present it to readers as an enhanced text. In addition, the impact of using the comprehension engine in improving the comprehension through evaluating the comprehension rate on participants and make a comparison between the two-phase results will be studied.

References

[1] R. G. Ortego and I. M. Sánchez, "Relevant parameters for the classification of reading books depending on the degree of textual readability in primary and compulsory secondary education (cse) students," IEEE Access, vol. 7, pp. 79044-79055, 2019.

[2] N. S. Al Madi, "A Study of Learning Performance and Cognitive Activity During Multimodal Comprehension Using Segmentation-integration Model and EEG," Kent State University, 2014.

[3] R. P. Carver, "Reading rate: Theory, research, and practical implications," Journal of Reading, vol. 36, no. 2, pp. 84-95, 1992.

[4] R. P. Carver, "Rauding theory predictions of amount comprehended under different purposes and speed reading conditions," Reading Research Quarterly, pp. 205-218, 1984.

[5] A. P. Widyassari, S. Rustad, G. F. Shidik, E. Noersasongko, A. Syukur, and A. Affandy, "Review of Automatic Text Summarization Techniques & Methods," Journal of King Saud University-Computer and Information Sciences, 2020.

[6] H. Van Lierde and T. W. Chow, "Learning with fuzzy hypergraphs: A topical approach to query-oriented text summarization," Information Sciences, vol. 496, pp. 212-224, 2019.

[7] Á. Hernández-Castañeda, R. A. García-Hernández, Y. Ledeneva, and C. E. Millán-Hernández, "Extractive Automatic Text Summarization Based on Lexical-Semantic Keywords," IEEE Access, vol. 8, pp. 49896-49907, 2020.

[8] M. N. Azadani, N. Ghadiri, and E. Davoodijam, "Graph-based biomedical text summarization: An itemset mining and sentence clustering approach," Journal of biomedical informatics, vol. 84, pp. 42-58, 2018.

[9] T. A. Fuad, M. T. Nayeem, A. Mahmud, and Y. Chali, "Neural sentence fusion for diversity driven abstractive multi-document summarization," Computer Speech & Language, vol. 58, pp. 216-230, 2019.

[10] J. Ding, Y. Li, H. Ni, and Z. Yang, "Generative Text Summary Based on Enhanced Semantic Attention and Gain-Benefit Gate," IEEE Access, vol. 8, pp. 92659-92668, 2020.

[11] L. Cagliero, L. Farinetti, and E. Baralis, "Recommending personalized summaries of teaching materials," IEEE Access, vol. 7, pp. 22729-22739, 2019.

[12] T. Yu, J. Li, Q. Yu, Y. Tian, X. Shun, L. Xu, L. Zhu, and H. Gao, "Knowledge graph for TCM health preservation: design, construction, and applications," Artificial Intelligence in Medicine, vol. 77, pp. 48-52, 2017.

[13] A. Babour, J. I. Khan, and F. Nafa, "Deepening Prose Comprehension by Incremental Free Text Conceptual Graph Mining and Knowledge," in 2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2016, pp. 208-215: IEEE.

[14] J. I. Khan and M. S. Hardas, "Does sequence of presentation matter in reading comprehension? a model based analysis of semantic concept network growth during reading," in 2013 IEEE Seventh International Conference on Semantic Computing, 2013, pp. 444-452: IEEE.

[15] M. Hart. (1971). Project Gutenberg. Available: http://www.gutenberg.org/, Accessed on: 21 July, 2017.

[16] A. Agrawal and A. An, "Unsupervised emotion detection from text using semantic and syntactic relations," in 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, 2012, vol. 1, pp. 346-353: IEEE.

[17] D. Chandran, K. Crockett, D. Mclean, and Z. Bandar, "FAST: A fuzzy semantic sentence similarity measure," in 2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2013, pp. 1-8: IEEE.

[18] H. Takahashi, "An approximate solution for the Steiner problem in graphs," 1980.

[19] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," Acta informatica, vol. 15, no. 2, pp. 141-145, 1981.

[20] A. Babour, "A Computational Mimicry of the Knowledge Augmentation Process in Comprehension based Learning," Kent State University, 2017.

[21] M. A. Ragab and A. Arisha, "Knowledge management and measurement: a critical review," Journal of knowledge management, 2013.

[22] C. E. Shannon and W. Weaver, The Mathematical Theory of Communication, by CE Shannon (and Recent Contributions to the Mathematical Theory of Communication), W. Weaver. University of illinois Press, 1949.

[23] R. Hekmat and P. Van Mieghem, "Study of connectivity in wireless ad-hoc networks with an improved radio model," in Proc. of WiOpt, 2004, vol. 10.

[24] P. Drieger, "Semantic network analysis as a method for visual text analytics," Procedia-social and behavioral sciences, vol. 79, no. 2013, pp. 4-17, 2013.

[25] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," nature, vol. 393, no. 6684, pp. 440-442, 1998.

[26] N. S. Al Madi and J. I. Khan, "Is learning by reading a book better than watching a movie? a computational analysis of semantic concept network growth during text and multimedia comprehension," in 2015 International Joint Conference on Neural Networks (IJCNN), 2015, pp. 1-8: IEEE.

[27] T. F. Coleman and J. J. Moré, "Estimation of sparse Jacobian matrices and graph coloring blems," SIAM journal on Numerical Analysis, vol. 20, no. 1, pp. 187-209, 1983.

[28] G. A. Miller, "WordNet: a lexical database for English," Communications of the ACM, vol. 38, no. 11, pp. 39-41, 1995.

[29] S. Menaka and N. Radha, "Text classification using keyword extraction technique," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 12, pp. 734-740, 2013.

[30] J.-M. Chen, M.-C. Chen, and Y. S. Sun, "A novel approach for enhancing student reading comprehension and assisting teacher assessment of literacy," Computers & Education, vol. 55, no. 3, pp. 1367-1382, 2010.

[31] J. Kamps, M. Marx, R. J. Mokken, and M. De Rijke, "Using WordNet to measure semantic orientations of adjectives," in LREC, 2004, vol. 4, pp. 1115-1118: Citeseer.

[32] A. Babour, F. Nafa, and J. I. Khan, "Connecting the dots in a concept space by Iterative reading of Freetext references with Wordnet," in 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), 2015, vol. 1, pp. 441-444: IEEE.

[33] Ethane, Encyclopaedia Britannica [Online], 2013, https://www.britannica. Com/science/ethane, Accessed on: July 24, 2017.

[34] K. Grens. (Mar 22, 2016). New Test for Zika OKed. Available: https://www.the-scientist.com/the-nutshell/new-test-for-zika-oked-33858, Accessed on: July 24, 2017.

[35] M. DeMarco. (April 7, 2015). Anesthesia gases are warming the planet. Available: https://www.sciencemag.org/news/2015/04/anesthesia-gases-are-warming-planet, Accessed on: July 24, 2017.

APPENDIX

LIST OF THE REFERENCE TEXT ARTICLES USED IN THE EXPERIMENT.

Title	Link
Ethane	https://en.wikipedia.org/w/index.php?title=Ethane&oldid=784178139
Hydrocarbon	https://en.wikipedia.org/w/index.php?title=Hydrocarbon&oldid=797145615
Hydrogen	https://en.wikipedia.org/w/index.php?title=Hydrogen&oldid=794824238
Carbon	https://en.wikipedia.org/w/index.php?title=Carbon&oldid=797606653
Chemical substance	https://en.wikipedia.org/w/index.php?title=Chemical_substance&oldid=797496319
Petroleum	https://en.wikipedia.org/w/index.php?title=Petroleum&oldid=797608302
Carbonization	https://en.wikipedia.org/w/index.php?title=Carbonization&oldid=795078223
Coal	https://en.wikipedia.org/w/index.php?title=Coal&oldid=797262457
Zika fever	https://en.wikipedia.org/w/index.php?title=Zika_fever&oldid=797239605%20130
Infection	https://en.wikipedia.org/w/index.php?title=Infection&oldid=797795424
Dengue fever	https://en.wikipedia.org/w/index.php?title=Dengue_fever&oldid=798056254
Chikungunya	https://en.wikipedia.org/w/index.php?title=Chikungunya&oldid=797661262
Virus	https://en.wikipedia.org/w/index.php?title=Virus&oldid=797985669
Aedes	https://en.wikipedia.org/w/index.php?title=Aedes&oldid=797888558
Mosquito	https://en.wikipedia.org/w/index.php?title=Mosquito&oldid=797321614
Antibody	https://en.wikipedia.org/w/index.php?title=Antibody&oldid=797925412 119
Anesthetic	https://en.wikipedia.org/w/index.php?title=Anesthetic&oldid=796835414
Climate	https://en.wikipedia.org/w/index.php?title=Climate&oldid=797528936
Oxide	https://en.wikipedia.org/w/index.php?title=Oxide&oldid=789414027
Desflurane	https://en.wikipedia.org/w/index.php?title=Desflurane&oldid=797545466
Isoflurane	https://en.wikipedia.org/w/index.php?title=Isoflurane&oldid=797545327 124
Sevoflurane	https://en.wikipedia.org/w/index.php?title=Sevoflurane&oldid=797688679
Halothane	https://en.wikipedia.org/w/index.php?title=Halothane&oldid=797545700